Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application.

1. (currently amended) A method for decreasing a computer application start-up time, comprising:

creating a serialized representation of application objects in a runtime environment;

building an optimized object code file using the serialized representation, wherein the optimized object code file replaces the initialization code compiled from the application source code; and

loading the optimized object code file into ~~providing the application to~~ a new runtime environment.

2. (currently amended) The method of claim 1 wherein the step of creating includes:

reading from a runtime memory space a description of each object of a running computer application.

3. (currently amended) The method of claim 1 wherein the step of creating includes enumerating a description of each object of ~~the application~~ the computer application using reflection.

4. (original) The method of claim 2 wherein the runtime environment is a virtual machine.

5. (currently amended) The method of claim 4 wherein the virtual machine is a ~~Flash~~ presentation renderer.

6. (currently amended) The method of claim 1 wherein the step of creating comprises:

identifying each object of a running computer application by a unique identifier.

7. (original) The method of claim 6 wherein the step of creating comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

8. (currently amended) The method of claim 6 wherein the step of creating includes the step of providing the serialized representation directly to the building step.

9. (currently amended) The method of claim 1 wherein the serialized representation of application objects in a runtime environment ~~application~~ is written in an ~~extensible markup language~~ Extensible Markup Language data format.

10. (currently amended) The method of claim 9 wherein the step of creating comprises storing a ~~the~~ markup language file prior to said step of building.

11. (original) The method of claim 1 wherein the step of creating comprises assigning a serialization identifier to each object.

12. (currently amended) The method of claim 1 further including the step of developing ~~the application~~ the computer application in an interpreted language.

13. (currently amended) The method of claim 6 wherein the step of creating ~~serializing~~ comprises:

assigning a function ID to each function in ~~the application~~ the computer application.

14. (original) The method of claim 13 wherein the method further includes:

creating a function ID table associating each function ID with function code.

15. (original) The method of claim 13 wherein the method further includes assigning unique function identifiers to functions within closures.

16. (currently amended) The method of claim 1 wherein the step of building comprises using a compiled version of application source code compiled prior to the creating step in combination with the serialized representation to build the new object code file.

17. (currently amended) The method of claim 1 wherein the step of creating comprises:

writing the serialized representation ~~description~~ to a text file prior to said step of building.

18. (canceled)

19. (original) The method of claim 1 wherein the step of creating is performed in a different runtime.

20. (currently amended) A method for providing an optimized application, comprising:

compiling an application provided in ~~an~~ a source language;

initializing the application in a runtime environment; and

creating a serialized representation of the application.

21. (currently amended) The method of claim 20 wherein the step of creating comprises:

reading from the runtime environment a description of each object of the ~~running~~ application.

22. (original) The method of claim 21 wherein the step of creating further comprises:

outputting the description to a rebuilder.

23. (currently amended) The method of claim 22 wherein the step of outputting comprises storing the serialized representation in a text file and providing the text file to the rebuilder.

24. (currently amended) The method of claim 22 wherein the step of outputting comprises:

writing the description to an Extensible Markup Language file and providing the Extensible Markup Language file to the rebuilder.

25. (original) The method of claim 20 wherein the runtime environment is a virtual machine.

26. (currently amended) The method of claim 25 ~~20~~ wherein the virtual machine is a ~~Flash~~ presentation renderer.

27. (original) The method of claim 20 wherein the step of creating includes assigning a serialization identifier to each initialized object.

28. (original) The method of claim 20 wherein the step of creating includes the steps of enumerating each object in a global scope and writing a serialized description of each said object.

29. (original) The method of claim 20 wherein the step of creating ~~serializing~~ comprises:

assigning a function ID to each function in the application.

30. (original) The method of claim 29 wherein the method further includes:

creating a function ID table associating each function ID with a function call.

31. (original) The method of claim 29 wherein the method further includes assigning function identifiers to functions within closures.

32. (canceled)

33. (currently amended) A method of operating an application, comprising:

requesting an application from an application source server; ~~and~~

receiving object code of a serialized description of the application from the application source server, the object code including instructions for replicating ~~creating~~ a runtime memory state of the application~~.~~; and

loading the object code received from the application source server into a runtime environment.

34. (original) The method of claim 33 wherein the object code includes media assets.

35. (currently amended) The method of claim 33 wherein the runtime memory state is of a ~~Flash~~ presentation renderer.

36. (currently amended) The method of claim 33 ~~wherein the step of receiving includes~~ further including the step of:

~~loading the object code into the runtime; and~~

executing the object code.

37. (cancelled)

38. (currently amended) One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising the steps of:

creating a serialized representation of application objects in a runtime environment;

building an optimized object code file using the serialized representation of the application objects and the compiled object code associated with the computer application, wherein the optimized object code file replaces the initialization code compiled from the application source code; and

loading the optimized object code file into ~~providing the application to~~ a new runtime environment.

39. (original) One or more processor readable storage devices as described in claim 38 wherein the step of creating includes:

reading from a runtime environment memory space a description of each object of a running application.

40. (original) One or more processor readable storage devices as described in claim 38 wherein the step of creating includes enumerating a description of each object of the application using reflection.

41. (original) One or more processor readable storage devices as described in claim 38 wherein the step of creating comprises:

identifying each object of a running application by a unique identifier.

42. (currently amended) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing ~~the~~ a description to a text file and compiling the text file.

43. (currently amended) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing ~~the~~ a description to an ~~Extensible markup language~~ Extensible Markup Language file and compiling a ~~the~~ markup language file.

44. (original) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

45. (currently amended) One or more processor readable storage devices as described in claim 41 wherein the ~~serialization~~ step of creating further includes the ~~step~~ steps of:

determining whether the object is a class; and

writing a serialized description of the class.

46. (currently amended) One or more processor readable storage devices as described in claim 39 wherein the serialized representation of application objects in a runtime environment ~~application~~ is written in an ~~extensible markup language~~ Extensible Markup Language data format.

47. (original) One or more processor readable storage devices as described in claim 39 wherein the step of creating comprises assigning a serialization identifier to each object.

48. (original) One or more processor readable storage devices as described in claim 39 further including the step of developing the application in an interpreted language.

49. (currently amended) One or more processor readable storage devices as described in claim 41 wherein the step of creating ~~serializing~~ comprises:

assigning a function ID to each function in the application.

50. (original) One or more processor readable storage devices as described in claim 49 wherein the method further includes:

creating a function ID table associating each function ID with function code.

51. (original) One or more processor readable storage devices as described in claim 49 wherein the method further includes assigning function identifiers to functions within closures.

52. (currently amended) A method of developing an application, comprising:

compiling first object code for an application;

loading the application into a first runtime environment;

creating a serialized representation of a memory space in said first runtime environment;

building a second object code using said serialized representation, wherein the second object code replaces the initialization code compiled from the application source code; and

~~deploying~~ loading said second object code into a second runtime environment.

53. (original) The method of claim 52 wherein the step of compiling is performed on an interpreted language application.

54. (original) The method of claim 52 wherein the step of creating is performed by calling at least one function from said first runtime environment.

55. (original) The method of claim 52 wherein the step of creating is performed in the same runtime environment as said application.

56. (original) The method of claim 52 wherein the step of creating is performed in a different runtime environment from said application.

57. (original) The method of claim 52 wherein the step of loading the application further includes executing portions of the application marked for execution prior to said creating step.

58. (currently amended) A method, comprising:

receiving from a runtime environment a serialized representation of objects in a memory space of the runtime environment; and

building an optimized object code file using the serialized representation and a compiled object code file used to create the memory space, wherein the optimized object code file replaces an initialization code compiled from the application source code.

59. (canceled)

60. (original) The method of claim 58 further including the step of initializing a serialization process in a separate memory space to create said serialized representation.

61. (original) The method of claim 58 further including the step of initializing a serialization process in the runtime environment to create said serialized representation.

62. (currently amended) A method for delivering an application via a network, comprising:

creating a serialized representation of application objects in a runtime environment;

building an optimized object code file using the serialized representation, wherein the optimized object code file replaces the initialization code compiled from the application source code; and

~~providing~~ loading the optimized object code file into ~~to~~ a new runtime environment via the network.

63. (currently amended) The method of claim 62 wherein the step of creating includes enumerating a description of each object of the application using reflection.

64. (original) The method of claim 63 wherein the new runtime environment is a virtual machine.

65. (currently amended) The method of claim 64 wherein the virtual machine is a ~~Flash~~ presentation renderer.